Provisioning Cloud-Based Computing Resources via a Dynamical Systems Approach

Marty Kandes

University of California, San Diego

May 18, 2016
Objective

Build a service for provisioning cloud-based computing resources that can be used to augment users’ existing, fixed resources and meet their batch job demands.
Vision
condor_annex = HTCondor + Amazon Web Services

condor_annex is a Perl-based script that utilizes the AWS CLI and other AWS services to orchestrate the delivery of HTCondor execute nodes from the cloud to your HTCondor pool.

Some key features:

- Supports bidding for spot instances.
- Instances sitting idle, not running user jobs will terminate after a fixed idle time (20 min).
- Each “annex” itself also has a finite lifetime.
My Problem

How many instances do I order with condor_annex to meet current user job demand?
My Original Assumptions

Known knowns:
- Idle instances terminate after a fixed lifetime (20min)
- Instances terminate when annex lease expires
- Assume (for now) single-core user jobs and instances

Known unknowns:
- User jobs arrive in queue at some unknown rate
- More user jobs than instances that can be purchased
- User jobs flock away to “free” resources at some unknown rate
- User job runtimes are unknown at submission
- Spot instances are preempted at some unknown rate
- Spot prices vary with time
Optimization Problem vs. Control Problem

- Forget optimally scheduling jobs and resources; too hard.
- Instead, seek to provision resources in a controlled way.
- Build a system that aims to use resources safely and efficiently.
Logistic Map: \( x_{n+1} = \sigma x_n (1 - x_n) \), where \( 0 \leq x_0 \leq 1 \).
An Oversimplified Provisioning Model

\[ \frac{dN}{dt} = \sigma N \left(1 - \frac{N}{K}\right) - \lambda N \]
Dynamical Systems 101

\[ \frac{dN}{dt} = f(N) = \sigma N \left( 1 - \frac{N}{K} \right) - \lambda N \]

1. **Find equilibria.** Set \( \frac{dN}{dt} = 0 \) and solve for \( N^* \).

\[
\sigma N^* \left( 1 - \frac{N^*}{K} \right) - \lambda N^* = 0 \implies N^* = 0, K \left( 1 - \frac{\lambda}{\sigma} \right)
\]

2. **Check stability of equilibria.**

\[
\frac{df}{dN} = \sigma - 2\sigma \frac{N}{K} - \lambda
\]

\[
\left. \frac{df}{dN} \right|_{N^* = 0} = \sigma - \lambda < 0 \iff \sigma < \lambda
\]

\[
\left. \frac{df}{dN} \right|_{N^* = K \left( 1 - \frac{\lambda}{\sigma} \right)} = \lambda - \sigma < 0 \iff \sigma > \lambda
\]
Provisioning Model I: State Diagram

- Spin-up
- Submit
- Flock
- Provision
- Match
- Terminate
- Complete
- Restart
- X_q
- X_R
- X_I
- X_Q
- Terminate
Provisioning Model I: System of Equations

\[
\frac{d x_q}{dt} = \Sigma_q - \sigma_{IR} x_q x_I - \sigma_{fq} x_q + \sigma_{Rq} x_R \\
\frac{d x_Q}{dt} = \sigma_{Qq} x_q - \sigma_{Ql} x_Q \\
\frac{d x_I}{dt} = \sigma_{Ql} x_Q - \sigma_{IR} x_q x_I + \sigma_{RI} x_R - \sigma_{IT} x_I \\
\frac{d x_R}{dt} = \sigma_{IR} x_q x_I - \sigma_{RI} x_R - \sigma_{Rq} x_R - \sigma_{RT} x_R
\]
Provisioning Model I: Definitions

- $x_q =$ number of user jobs in the queue
- $x_Q =$ number of instances in the queue
- $x_I =$ number of instances sitting idle
- $x_R =$ number of instances busy running user jobs
- $\Sigma_q =$ rate of user job submission (jobs/time)
- $\sigma_{IR} = 1/\tau_{IR} =$ matchmaking rate; $\tau_{IR} =$ idle-running lifetime
- $\sigma_{qf} = 1/\tau_{qf} =$ flocking rate; $\tau_{qf} =$ flocking lifetime
- $\sigma_{Rq} = 1/\tau_{Rq} =$ restart rate; $\tau_{Rq} =$ restart lifetime
- $\sigma_{qQ} =$ queueing rate
- $\sigma_{QI} = 1/\tau_{QI} =$ instance spin-up rate; $\tau_{QI} =$ annex start-up time
- $\sigma_{RI} = 1/\tau_{RI} =$ job completion rate; $\tau_{RI} =$ job lifetime
- $\sigma_{IT} = 1/\tau_{IT} =$ idle termination rate; $\tau_{IT} =$ idle-termination lifetime
- $\sigma_{RT} = 1/\tau_{RT} =$ running termination rate; $\tau_{RT} =$ annex lifetime
Provisioning Model I: Equilibria

Solve.

\[
\begin{align*}
\frac{dx_q}{dt} &= f_q(x_q, x_Q, x_I, x_R) = 0 \\
\frac{dx_Q}{dt} &= f_Q(x_q, x_Q, x_I, x_R) = 0 \\
\frac{dx_I}{dt} &= f_I(x_q, x_Q, x_I, x_R) = 0 \\
\frac{dx_R}{dt} &= f_R(x_q, x_Q, x_I, x_R) = 0
\end{align*}
\]

Find two equilibrium points.

\[
\begin{align*}
x_1^* &= (x_{q1}^*, x_{Q1}^*, x_{I1}^*, x_{R1}^*) \\
x_2^* &= (x_{q2}^*, x_{Q2}^*, x_{I2}^*, x_{R2}^*)
\end{align*}
\]
Provisioning Model I: Stability of Equilibria

Find Jacobian.

\[
J = \frac{df}{dx} = \begin{bmatrix}
\frac{df_q}{dx_q} & \frac{df_q}{dx_Q} & \frac{df_q}{dx_I} & \frac{df_q}{dx_R} \\
\frac{df_q}{dx_q} & \frac{df_q}{dx_Q} & \frac{df_q}{dx_I} & \frac{df_q}{dx_R} \\
\frac{df_q}{dx_q} & \frac{df_q}{dx_Q} & \frac{df_q}{dx_I} & \frac{df_q}{dx_R} \\
\frac{df_q}{dx_q} & \frac{df_q}{dx_Q} & \frac{df_q}{dx_I} & \frac{df_q}{dx_R}
\end{bmatrix}
\]

Compute eigenvalues of Jacobian about \( x_1^* \) and \( x_2^* \).

\[f(x) = f(x^*) + J(x^*)(x - x^*) + \cdots\]

If the eigenvalues all have real parts that are negative, then the system is **stable** near the stationary point, if any eigenvalue has a real part that is positive, then the point is **unstable**.
Validation Test I: Parameters

- $x_q(t = 0) = x_Q(t = 0) = x_I(t = 0) = x_R(t = 0) = 0$
- $\Sigma_q = 60$ jobs per hour
- $\sigma_{IR} = 1 / \tau_{IR} = 1 / 5$ minutes
- $\sigma_{qf} = 0$ (No flocking)
- $\sigma_{Rq} = 0$ (No restarts)
- $\sigma_{qQ} = 0.1$
- $\sigma_{QI} = 1 / \tau_{QI} = 1 / 10$ minutes
- $\sigma_{RI} = 1 / \tau_{RI} = 1 / 2$ hours
- $\sigma_{IT} = 1 / \tau_{IT} = 1 / 20$ minutes
- $\sigma_{RT} = 1 / \tau_{RT} = 1 / 12$ hours
- $x_1^* = (-1.71566, -0.0285943, 2.91433, 102.857)$
- $x_2^* = (87.4299, 1.45717, 0.0571886, 102.857)$
- $\lambda_1 = (54.4891, -5.9492, -1.98, -0.583333)$
- $\lambda_2 = (-1052.84, -5.89802, -0.583333, -0.103362)$
Validation Test I: Simulation Results (72 Hours)
Validation Test I: Experimental Results (72 Hours)
Possible Source of Oscillations

Discretization-induced (discrete time, discrete state)

Delay-induced (discrete delay); *Hopf bifurcation*
New “Large Workflow” Assumptions

Provision resources based on individual submissions

\[ N = \text{jobs per user submission} \gg M = \text{max instances} \]

User-specified workflow “deadline”

\[ T_{\text{deadline}} \gg \tau_{RT} > \tau_{RI} > \Delta t \]

User-specified estimate of average job lifetime, \( \tau_{RI} \).

Meet deadline or run out of money; minimize waste and cost
Acknowledgments

Todd Miller @ UW - Madison
Center for High Throughput Computing, HTCondor

Frank Würthwein @ UCSD
Open Science Grid, Executive Director

Jeffery Dost @ UCSD
Open Science Grid, Glidein Factory Operations

Edgar Fajardo @ UCSD
Open Science Grid, Software
Questions?